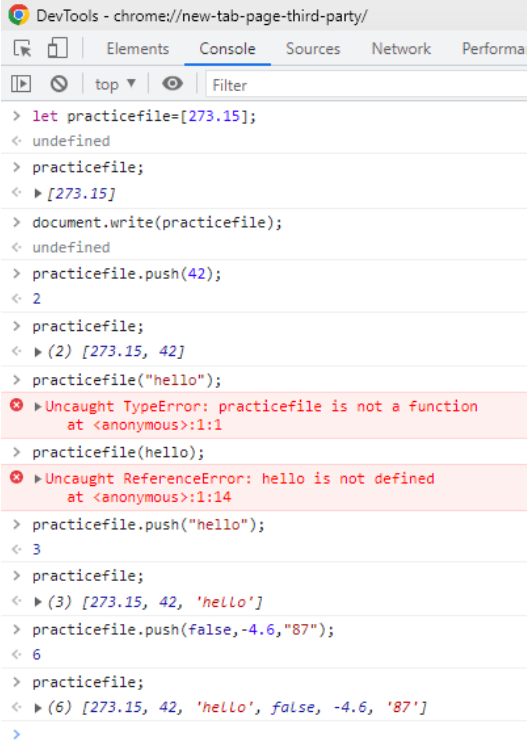
SDE TASK-3

1. Create an array called practiceFile with the following entry: 273.15. Use the push method to add the following elements to the array. Add items a & b one at a time, then use a single push to add the items in part c. Print the array after each step to confirm the changes.

a. 42
b. "hello"
c. false, -4.6, "87"

SOLUTION:

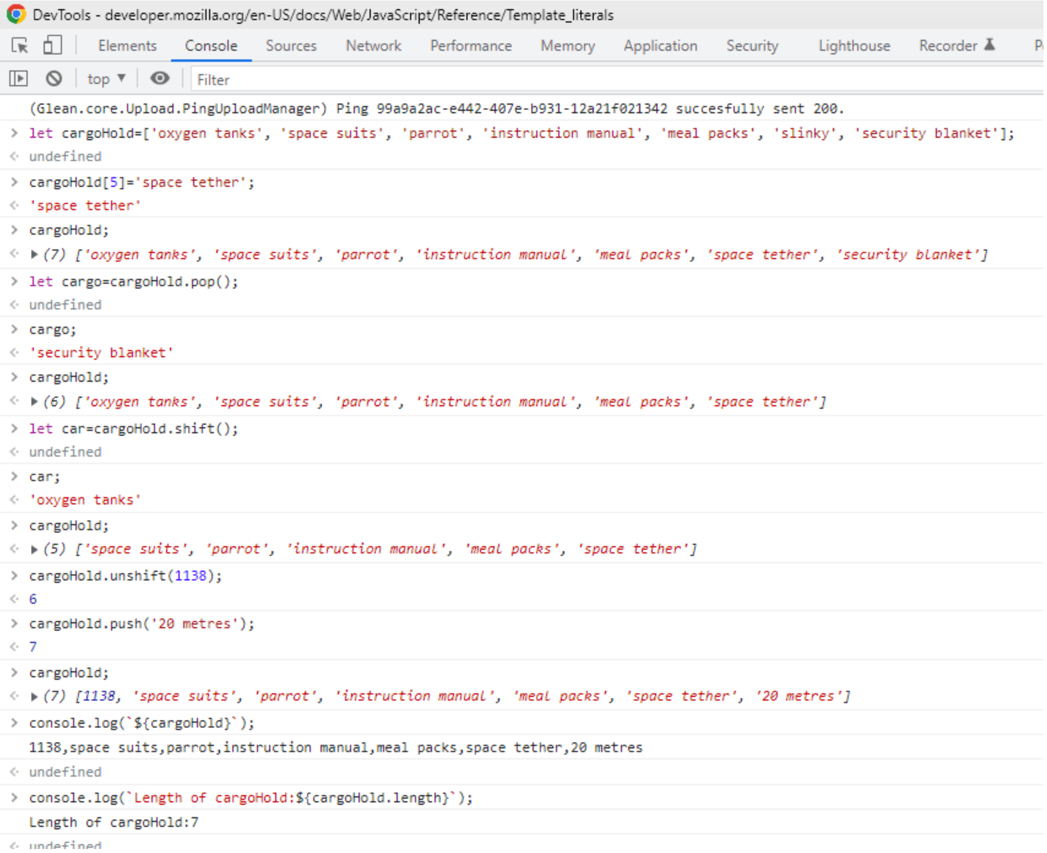


```
> let practicefile=[273.15];
<· undefined
> practicefile;
<· ▶[273.15]
> document.write(practicefile);
<· undefined
> practicefile.push(42);
<· 2
> practicefile;
<· ▶(2) [273.15, 42]
> practicefile("hello");
⊗ ▶Uncaught TypeError: practicefile is not a function
       at <anonymous>:1:1
> practicefile(hello);
⊗ ▶Uncaught ReferenceError: hello is not defined
       at <anonymous>:1:14
> practicefile.push("hello");
<· 3
> practicefile;
<· ▶(3) [273.15, 42, 'hello']
> practicefile.push(false,-4.6,"87");
<· 6
> practicefile;
<· ▶(6) [273.15, 42, 'hello', false, -4.6, '87']
>
```

2. push, pop, shift and unshift are used to add/remove elements from the beginning/end of an array. Bracket notation can be used to modify any element within an array. Starting with the cargoHold array ['oxygen tanks', 'space suits', 'parrot', 'instruction manual', 'meal packs', 'slinky', 'security blanket'], write statements to do the following:
a. Use bracket notation to replace 'slinky' in the array with 'space tether'. Print the array to confirm the change.
b. Remove the last item from the array with pop. Print the element removed and the updated array.
c. Remove the first item from the array with shift. Print the element removed and the updated array.

d. Unlike pop and shift, push and unshift require arguments inside the (). Add the items 1138 and '20 meters' to the array - the number at the start and the string at the end. Print the updated array to confirm the changes.
e. Use a template literal to print the final array and its length.

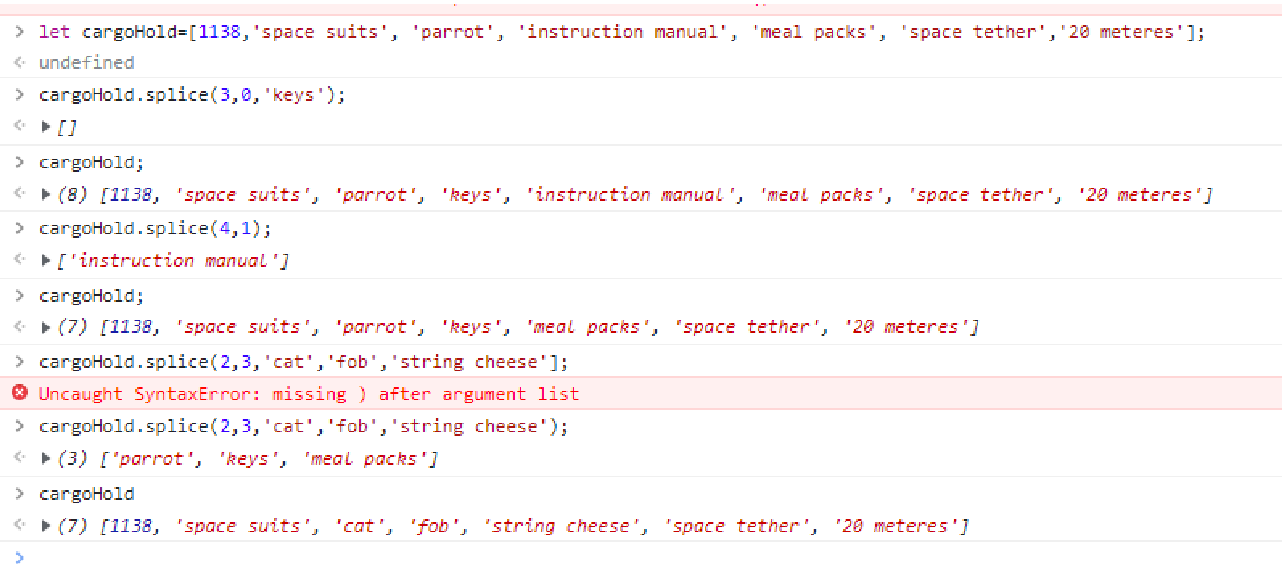
SOLUTION:



```
   (Glean.core.Upload.PingUploadManager) Ping 99a9a2ac-e442-407e-b931-12a21f021342 succesfully sent 200.
>  let cargoHold=['oxygen tanks', 'space suits', 'parrot', 'instruction manual', 'meal packs', 'slinky', 'security blanket'];
<· undefined
>  cargoHold[5]='space tether';
<· 'space tether'
>  cargoHold;
<· ▶(7) ['oxygen tanks', 'space suits', 'parrot', 'instruction manual', 'meal packs', 'space tether', 'security blanket']
>  let cargo=cargoHold.pop();
<· undefined
>  cargo;
<· 'security blanket'
>  cargoHold;
<· ▶(6) ['oxygen tanks', 'space suits', 'parrot', 'instruction manual', 'meal packs', 'space tether']
>  let car=cargoHold.shift();
<· undefined
>  car;
<· 'oxygen tanks'
>  cargoHold;
<· ▶(5) ['space suits', 'parrot', 'instruction manual', 'meal packs', 'space tether']
>  cargoHold.unshift(1138);
<· 6
>  cargoHold.push('20 metres');
<· 7
>  cargoHold;
<· ▶(7) [1138, 'space suits', 'parrot', 'instruction manual', 'meal packs', 'space tether', '20 metres']
>  console.log(`${cargoHold}`);
   1138,space suits,parrot,instruction manual,meal packs,space tether,20 metres
<· undefined
>  console.log(`Length of cargoHold:${cargoHold.length}`);
   Length of cargoHold:7
<· undefined
```

3.The **splice** method can be used to either add or remove items from an array. It can also accomplish both tasks at the same time. Use splice to make the following changes to the final cargoHold array from exercise 2. Be sure to print the array after each step to confirm your updates.
a. Insert the string 'keys' at index 3 without replacing any other entries.
b. Remove 'instruction manual' from the array. (Hint: indexOf is helpful to avoid manually counting an index).
c. Replace the elements at indexes 2 - 4 with the items 'cat', 'fob', and 'string cheese'.

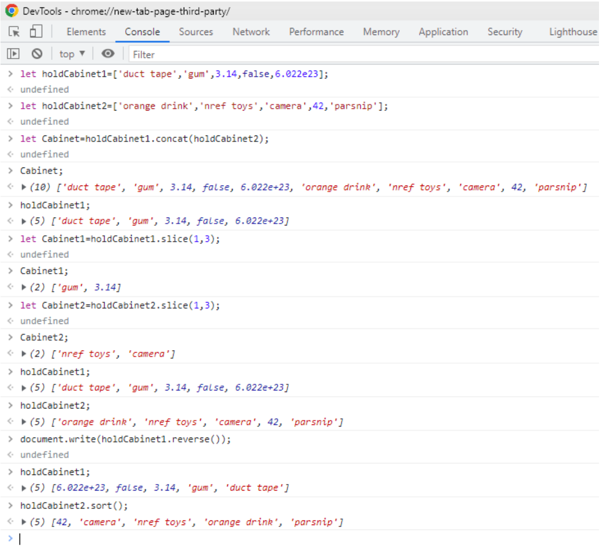
SOLUTION:



```
> let cargoHold=[1138,'space suits', 'parrot', 'instruction manual', 'meal packs', 'space tether','20 meteres'];
<· undefined
> cargoHold.splice(3,0,'keys');
<· ▸ []
> cargoHold;
<· ▸ (8) [1138, 'space suits', 'parrot', 'keys', 'instruction manual', 'meal packs', 'space tether', '20 meteres']
> cargoHold.splice(4,1);
<· ▸ ['instruction manual']
> cargoHold;
<· ▸ (7) [1138, 'space suits', 'parrot', 'keys', 'meal packs', 'space tether', '20 meteres']
> cargoHold.splice(2,3,'cat','fob','string cheese'];
⊗ Uncaught SyntaxError: missing ) after argument list
> cargoHold.splice(2,3,'cat','fob','string cheese');
<· ▸ (3) ['parrot', 'keys', 'meal packs']
> cargoHold
<· ▸ (7) [1138, 'space suits', 'cat', 'fob', 'string cheese', 'space tether', '20 meteres']
>
```

4. Some methods---like splice and push---alter the original array, while others do not. Use the arrays holdCabinet1 ['duct tape', 'gum', 3.14, false, 6.022e23] and holdCabinet2 ['orange drink', 'nerf toys', 'camera', 42, 'parsnip'] to explore the following methods: concat, slice, reverse, sort. Refer back to the chapter if you need to review the proper syntax for any of these methods.
a. Print the result of using concat on the two arrays. Does concat alter the original arrays? Verify this by printing holdCabinet1 after using the method.
b. Print a slice of two elements from each array. Does slice alter the original arrays?
c. reverse the first array, and sort the second. What is the difference between these two methods? Do the methods alter the original arrays?

SOLUTION:

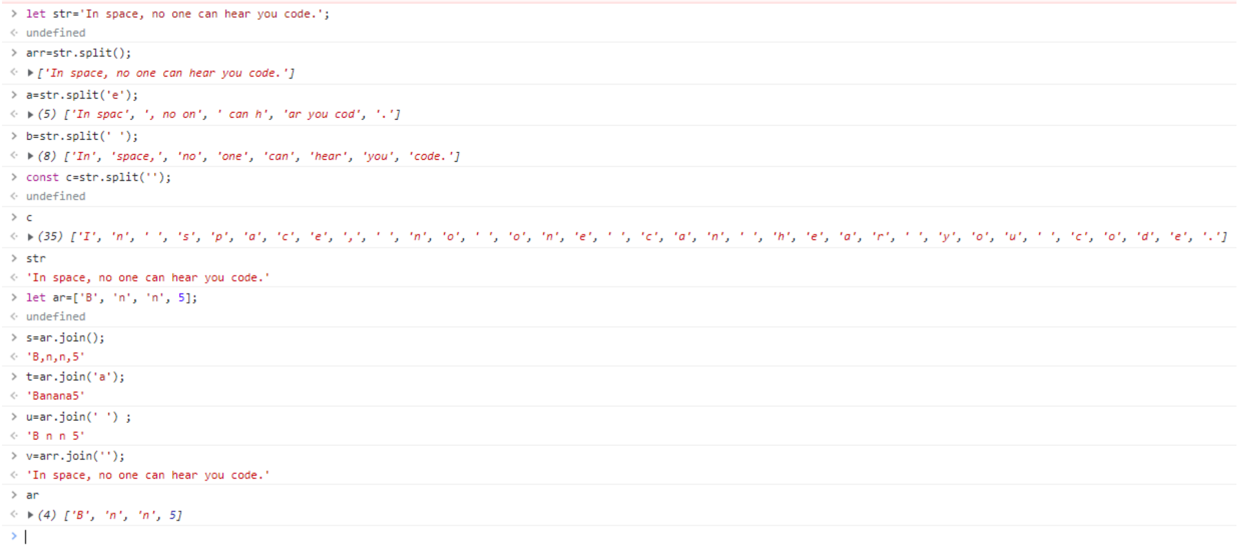


```
> let holdCabinet1=['duct tape','gum',3.14,false,6.022e23];
<· undefined
> let holdCabinet2=['orange drink','nref toys','camera',42,'parsnip'];
<· undefined
> let Cabinet=holdCabinet1.concat(holdCabinet2);
<· undefined
> Cabinet;
<· ▸ (10) ['duct tape', 'gum', 3.14, false, 6.022e+23, 'orange drink', 'nref toys', 'camera', 42, 'parsnip']
> holdCabinet1;
<· ▸ (5) ['duct tape', 'gum', 3.14, false, 6.022e+23]
> let Cabinet1=holdCabinet1.slice(1,3);
<· undefined
> Cabinet1;
<· ▸ (2) ['gum', 3.14]
> let Cabinet2=holdCabinet2.slice(1,3);
<· undefined
> Cabinet2;
<· ▸ (2) ['nref toys', 'camera']
> holdCabinet1;
<· ▸ (5) ['duct tape', 'gum', 3.14, false, 6.022e+23]
> holdCabinet2;
<· ▸ (5) ['orange drink', 'nref toys', 'camera', 42, 'parsnip']
> document.write(holdCabinet1.reverse());
<· undefined
> holdCabinet1;
<· ▸ (5) [6.022e+23, false, 3.14, 'gum', 'duct tape']
> holdCabinet2.sort();
<· ▸ (5) [42, 'camera', 'nref toys', 'orange drink', 'parsnip']
> |
```

- **Concat** and **slice** function does not affect original array.
- **Reverse** and **sort** function will affect original array.

5. The split method converts a string into an array, while the join method does the opposite.
a. Try it! Given the string str = 'In space, no one can hear you code.', see what happens when you print str.split() vs. str.split('e') vs. str.split(' ') vs. str.split('').
What is the purpose of the parameter inside the ()?
b. Given the array arr = ['B', 'n', 'n', 5], see what happens when you print arr.join() vs. arr.join('a') vs. arr.join(' ') vs. arr.join('').
What is the purpose of the parameter inside the ()?
c. Do split or join change the original string/array?
d. The benefit, cadet, is that we can take a string with delimiters (like commas)and convert it into a modifiable array.
Alphabetize these hold contents:"water,space suits,food,plasma sword,batteries", and then combine them into a new string.

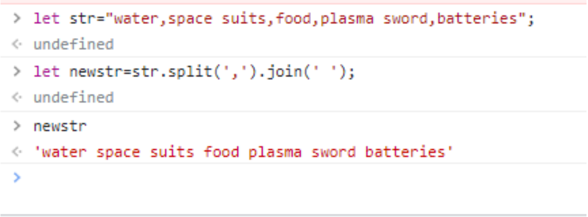
Solution for a & b:



```
> let str='In space, no one can hear you code.';
<· undefined
> arr=str.split();
<· ▶['In space, no one can hear you code.']
> a=str.split('e');
<· ▶(5) ['In spac', ', no on', ' can h', 'ar you cod', '.']
> b=str.split(' ');
<· ▶(8) ['In', 'space,', 'no', 'one', 'can', 'hear', 'you', 'code.']
> const c=str.split('');
<· undefined
> c
<· ▶(35) ['I', 'n', ' ', 's', 'p', 'a', 'c', 'e', ',', ' ', 'n', 'o', ' ', 'o', 'n', 'e', ' ', 'c', 'a', 'n', ' ', 'h', 'e', 'a', 'r', ' ', 'y', 'o', 'u', ' ', 'c', 'o', 'd', 'e', '.']
> str
<· 'In space, no one can hear you code.'
> let ar=['B', 'n', 'n', 5];
<· undefined
> s=ar.join();
<· 'B,n,n,5'
> t=ar.join('a');
<· 'Banana5'
> u=ar.join(' ') ;
<· 'B n n 5'
> v=arr.join('');
<· 'In space, no one can hear you code.'
> ar
<· ▶(4) ['B', 'n', 'n', 5]
> |
```

c.**split** and **join** function doesn't affect orginal array and string.

Solution for d:



```
> let str="water,space suits,food,plasma sword,batteries";
<· undefined
> let newstr=str.split(',').join(' ');
<· undefined
> newstr
<· 'water space suits food plasma sword batteries'
>
```

6. Arrays can hold different data types, even other arrays! A multi-dimensional array is one with entries that are themselves arrays.

a. Define and initialize the following arrays, which hold the name, chemical symbol and mass for different elements:

i. element1 = ['hydrogen', 'H', 1.008]

ii. element2 = ['helium', 'He', 4.003]

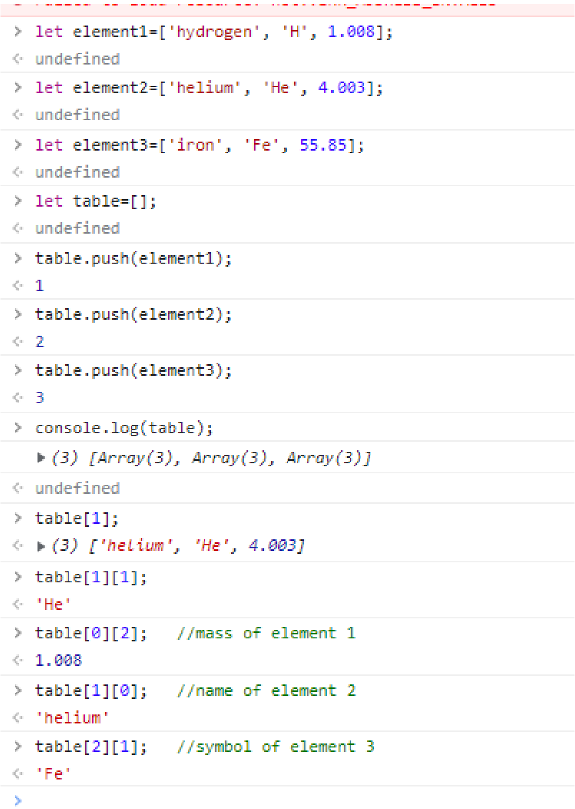
iii. element26 = ['iron', 'Fe', 55.85]

b. Define the array table, and use push(array Name) to add each of the element arrays to it. Print table to see its structure.

c. Use bracket notation to examine the difference between printing table[1] and table[1][1]. Don't just nod your head! I want to HEAR you describe this difference. Go ahead, talk to your screen.

d. Using bracket notation and the table array, print the mass of element1, the name for element 2 and the symbol for element26.

SOLUTION:



```
> let element1=['hydrogen', 'H', 1.008];
<· undefined
> let element2=['helium', 'He', 4.003];
<· undefined
> let element3=['iron', 'Fe', 55.85];
<· undefined
> let table=[];
<· undefined
> table.push(element1);
<· 1
> table.push(element2);
<· 2
> table.push(element3);
<· 3
> console.log(table);
   ▶(3) [Array(3), Array(3), Array(3)]
<· undefined
> table[1];
<· ▶(3) ['helium', 'He', 4.003]
> table[1][1];
<· 'He'
> table[0][2];   //mass of element 1
<· 1.008
> table[1][0];   //name of element 2
<· 'helium'
> table[2][1];   //symbol of element 3
<· 'Fe'
>
```

7. For each of the exercises below, assume you are starting with the following people array.

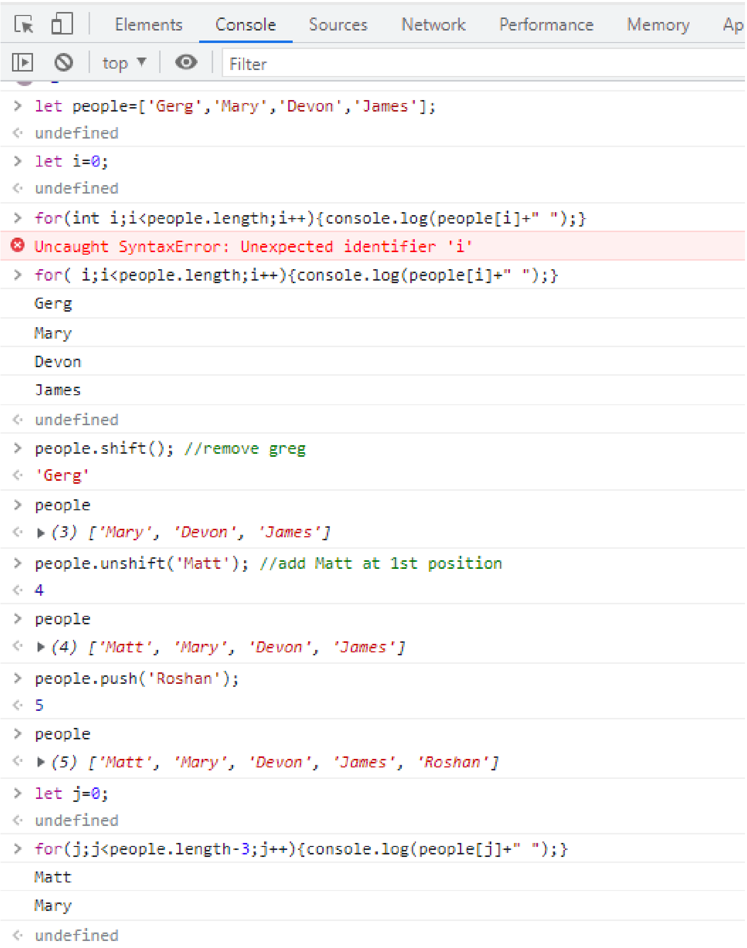
let people = ["Greg", "Mary", "Devon", "James"];

1.Using a loop, iterate through this array and console.log all of the people.

2.Write the command to remove "Greg" from the array.

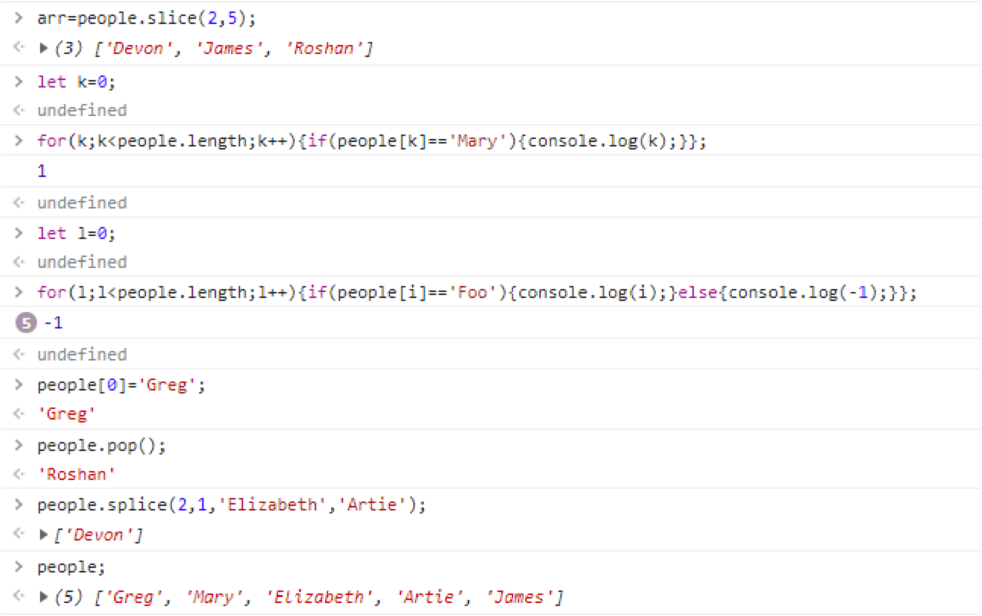
3.Write the command to remove "James" from the array.
4.Write the command to add "Matt" to the front of the array.
5.Write the command to add your name to the end of the array.
6.Using a loop, iterate through this array and after console.log-ing "Mary", exit from the loop.
7.Write the command to make a copy of the array using slice. The copy should NOT include "Mary" or "Matt".
8.Write the command that gives the indexOf where "Mary" is located.
9.Write the command that gives the indexOf where "Foo" is located (this should return -1).
10.Redefine the people variable with the value you started with. Using the splice command, remove "Devon" from the array and add "Elizabeth" and "Artie". Your array should look like this when you are done ["Greg", "Mary", "Elizabeth", "Artie",
"James"].
11.Create a new variable called withBob and set it equal to the people array concatenated with the string of "Bob".

Solution for 1-6:



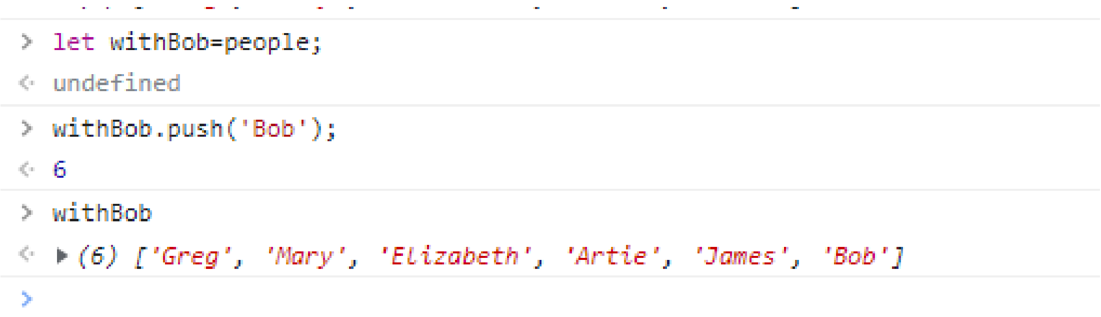
```
> let people=['Gerg','Mary','Devon','James'];
< undefined
> let i=0;
< undefined
> for(int i;i<people.length;i++){console.log(people[i]+" ");}
✖ Uncaught SyntaxError: Unexpected identifier 'i'
> for( i;i<people.length;i++){console.log(people[i]+" ");}
  Gerg
  Mary
  Devon
  James
< undefined
> people.shift(); //remove greg
< 'Gerg'
> people
< ▶(3) ['Mary', 'Devon', 'James']
> people.unshift('Matt'); //add Matt at 1st position
< 4
> people
< ▶(4) ['Matt', 'Mary', 'Devon', 'James']
> people.push('Roshan');
< 5
> people
< ▶(5) ['Matt', 'Mary', 'Devon', 'James', 'Roshan']
> let j=0;
< undefined
> for(j;j<people.length-3;j++){console.log(people[j]+" ");}
  Matt
  Mary
< undefined
```

Solution for 7 & 10:



```
> arr=people.slice(2,5);
<· ▶ (3) ['Devon', 'James', 'Roshan']
> let k=0;
<· undefined
> for(k;k<people.length;k++){if(people[k]=='Mary'){console.log(k);}};
  1
<· undefined
> let l=0;
<· undefined
> for(l;l<people.length;l++){if(people[i]=='Foo'){console.log(i);}else{console.log(-1);}};
5 -1
<· undefined
> people[0]='Greg';
<· 'Greg'
> people.pop();
<· 'Roshan'
> people.splice(2,1,'Elizabeth','Artie');
<· ▶ ['Devon']
> people;
<· ▶ (5) ['Greg', 'Mary', 'Elizabeth', 'Artie', 'James']
```

Solution 11:



```
> let withBob=people;
<· undefined
> withBob.push('Bob');
<· 6
> withBob
<· ▶ (6) ['Greg', 'Mary', 'Elizabeth', 'Artie', 'James', 'Bob']
>
```